UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Aplicación para la gestión y uso de escalas de evaluación en el ámbito sanitario.

Manuel Egido González
Tutor: Miguel Ángel Mora Rincón

JUNIO 2018

Aplicación para la gestión y uso de escalas de evaluación en el ámbito sanitario.

AUTOR: Manuel Egido González
TUTOR: Miguel Ángel Mora Rincón

Grupo de la EPS
Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2018

Resumen (castellano)

Es un hecho que las Tecnologías de la Información y la Comunicación están cambiando el mundo en el que vivimos y la forma en que se llevan a cabo la mayoría de las actividades y procesos.

Como no podía ser de otra forma, el sector sanitario está sufriendo una transformación tecnológica que permite una mejora sustancial en el tratamiento y evaluación de los pacientes, así como una mayor eficiencia en los procesos médicos. Esta transformación está siendo más lenta que en la mayoría de los sectores dada su complejidad, pero, a pesar de ello, es uno de los campos que mayor recorrido tendrá en los próximos años.

Este Trabajo de Fin de grado consiste en el desarrollo de una aplicación web que permita la evaluación de pacientes del ámbito sanitario.

La aplicación desarrollada ofrece la posibilidad de crear tu propia escala de evaluación para, posteriormente, evaluar a un paciente y guardar su diagnóstico en la base de datos del centro médico correspondiente. Además, permitirá compartir las escalas creadas con el resto de los usuarios de la aplicación.

Con esto, se pretende facilitar el trabajo a los profesionales sanitarios, que actualmente deben dedicar una gran parte de su trabajo a tareas ajenas a su especialidad, como es el paso de datos de formato físico a digital y, por otro lado, promover el intercambio de conocimiento mediante el intercambio de escalas de evaluación.

Este proyecto se centrará en obtener una aplicación intuitiva, sencilla y que pueda ser ejecutada en el mayor número posible de dispositivos. Con esto se busca llegar al mayor número de usuarios posibles.

Abstract (English)

It is a fact that Information and Communication Technologies are changing the world in which we live and the way in which most of the activities and processes are carried out.

As it could not be otherwise, the health sector is undergoing a technological transformation that allows a substantial improvement in the treatment and evaluation of patients, as well as greater efficiency in medical processes. This transformation is being slower than in most sectors given its complexity, but, despite this, it is one of the areas that will have the longest journey in the coming years.

This Bachelor Thesis consists in the development of a web application that allows the evaluation of patients in the health field.

The application developed offers the possibility of creating your own evaluation scale to, subsequently, evaluate a patient and save your diagnosis in the corresponding medical center database. In addition, it will allow sharing the scales created with the rest of the application users.

With this, it is intended to facilitate the work to health professionals, who currently must devote a large part of their work to tasks unrelated to their specialty, such as the transformation of data from physical format to digital and, on the other hand, promote the exchange of knowledge through the exchange of evaluation scales.

This project will focus on obtaining an intuitive and simple application that can be executed on as many devices as possible. This seeks to reach as many users as possible.

Palabras clave (castellano)

Escalas de Evaluación, Sanidad, Evaluación de Pacientes, Aplicación Web, Angular.

Keywords (inglés)

Evaluation Scales, Health, Patients Evaluation, Web Application, Angular, Bootstrap

Agradecimientos

A aquellos profesores que se preocuparon por mi a lo largo de estos años. En especial a mi tutor académico, José Dorronsoro, que me ayudó en uno de los momentos más delicados de mi estancia en la universidad.

A mi tutor de TFG Miguel Ángel Mora, que me ha guiado en el desarrollo de este proyecto y me ha ayudado a salir de muchos de callejones que parecian no tener salida.

A todos los compañeros que me han acompañado estos años, en especial a Adrián Galán, que cambió mi día a día en la facultad.

A mis padres, por su paciencia y por no dejar de confiar en mí en ningún momento.

Y en especial, a ti, Judith, la persona más especial de mi vida. Porque siempre que me caí, me levantaste. Sé, sin ninguna duda, que esto no hubiera sido posible sin ti.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

Este trabajo surge, como muchas buenas ideas, de una necesidad.

El sector sanitario está en continuo desarrollo, pero todavía hay procesos completamente ineficientes que producen un impacto directo en el trabajo llevado a cabo por los profesionales.

Al tener contacto directo con estos profesionales y tratar de entender su trabajo, me doy cuenta de un procedimiento que ocupa una gran parte de su trabajo, cuando podría ser realizado con solo pulsar un botón. En concreto, me refiero al tratamiento de los datos recogidos por los especialistas al evaluar a un paciente.

Para realizar el trabajo, me reuno con varios profesionales sanitarios, con Terapeutas Ocupacionales de las especialidades de Daño Cerebral, Salud Mental y Geriatría. Todos ellos pasan alrededor de un 20% de su jornada laboral, pasando sus evaluaciones a la base de datos del centro en el que trabajan.

Estas evaluaciones se llevan a cabo en formato físico para, posteriormente, pasarlas a formato digital y así poder guardarlas en la base de datos pertinente para que el resto del centro pueda acceder a ellas de una manera sencilla.

Por todo esto, decido implementar una aplicación web que ahorre todo ese proceso a los profesionales, de tal manera que, con algún dispositivo móvil o fijo, puedan evaluar a los pacientes y directamente enviar la información a la base datos. Además, siendo consciente de la gran variedad de campos y escalas de evaluación que se emplean hoy en el mundo sanitario, se dará la posibilidad al usuario de definir sus propias escalas y publicarlas para compartirlas con el resto de profesionales.

1.2 Objetivos

El objetivo principal de este trabajo es mejorar el proceso de recogida de datos por parte de los profesionales sanitarios.

La aplicación desarrollada dará la posibilidad de evaluar pacientes mediante escalas de evaluación. Una vez se evalúa al paciente, la aplicación deberá mostrar también el resultado de la evaluación. Además, la aplicación ofrecerá un apartado para la creación de escalas de evaluación y, una vez creadas, el usuario podrá compartirlas con el resto de la comunidad de usuarios de la aplicación o con algún usuario en concreto.

De esta manera, se conseguirá un mayor grado de eficiencia en el trabajo de los profesionales, sin perder tiempo en asuntos administrativos y, mediante el intercambio de escalas de evaluación, se pretende que los profesionales descubran nuevas escalas que puedan ser útiles para su trabajo y sus pacientes.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Capitulo 2**: **Estado del Arte.** En este capítulo se expone el trabajo de investigación realizado en lo referente al estudio de tecnologías y el porqué de las tecnologías empleadas en el desarrollo del trabajo.
- **Capitulo 3**: **Diseño.** Apartado enfocado a la descripción del Diseño de la aplicación implementada.
- **Capitulo 4**: **Desarrollo.** Se exponen todos los aspectos del desarrollo llevados a cabo, desde los lenguajes de programación y frameworks hasta la temporalidad de cada parte del desarrollo.
- **Capitulo 5**: **Integración, Pruebas y Resultados.** Se detallan las actividades realizadas para la evaluación del funcionamiento de la aplicación.
- **Capitulo 6**: **Conclusión y trabajo futuro.** Contiene las reflexiones abstraídas del desarrollo del proyecto así como qué se puede añadir y mejorar al proyecto.

2 Estado del arte

2.1 Estudio de Tecnologías

Para el desarrollo de la aplicación, se realizó un estudio de todas las tecnologías usadas actualmente en el ámbito del desarrollo de aplicaciones web.

El mundo del desarrollo web está sufriendo una de las evoluciones más acusadas del sector. Cada día surgen nuevas herramientas que dejan obsoletas otras con meses de vida. Esto hace que el estudio de tecnologías de este trabajo sea uno de los puntos más relevantes para su desarrollo.

Se comenzó con la división del desarrollo en dos partes. desarrollo Front-End y Back-End. Se define el desarrollo Front-End como aquel que interactúa con el usuario y recopila los datos pertinentes, mientras que el desarrollo Back-End será el encargado de manipular y procesar esos datos.

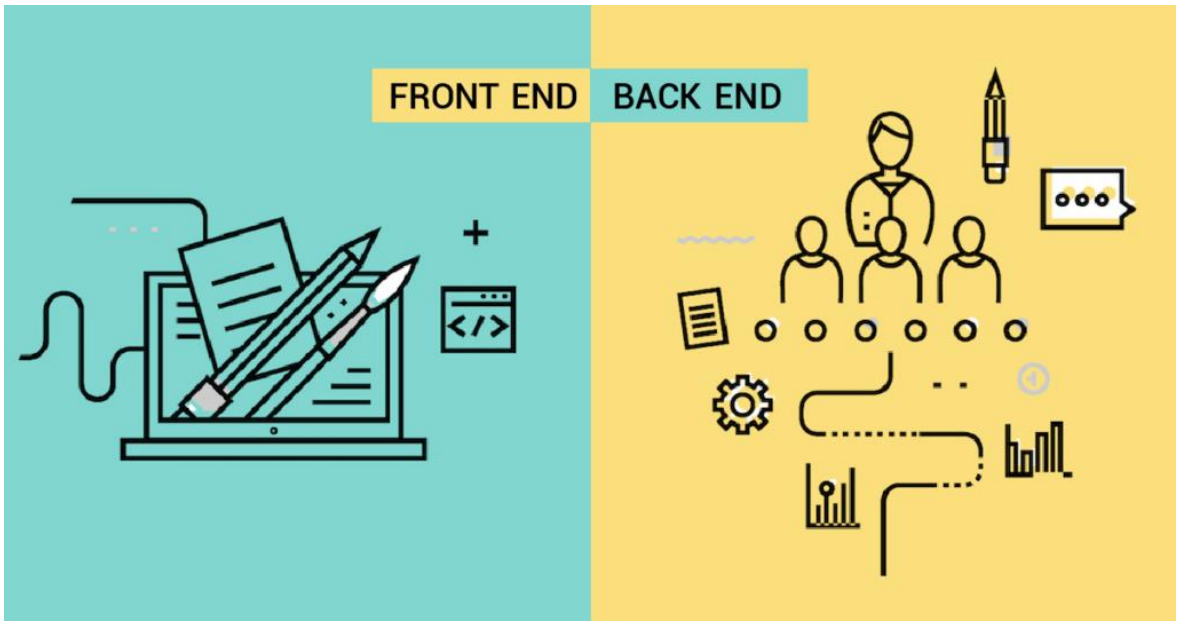

Figura 1: Frontend y Backend

A continuación, se detalla el estudio llevado a cabo de cada uno de los componentes que forman el proyecto.

2.1.1 Frameworks

Para el desarrollo de la aplicación, se comenzó con la elección de un framework para la parte Front-End

Un framework es una herramienta que adaptamos en nuestro proyecto para conseguir un desarrollo estructurado, robusto y más rápido.

Una de las principales ventajas de los frameworks en el desarrollo Front-End, es que ofrecen funcionalidades ya implementadas, y no es necesario codificarlas por el desarrollador. Ésto facilita y acelera notablemente el trabajo.

Para el desarrollo del proyecto se realizó un estudio de los frameworks más utilizados hoy en día por los expertos y las empresas tecnológicas, y nos centramos en 4:

2.1.1.1 Angular

Angular es un framework de código abierto desarrollado por Google. Es usado para el desarrollo de aplicacion SPA(Single Page Aplication).

Su lanzamiento se produjo en 2016 bajo el nombre de AngularJs, para, en 2017, cambiar su nombre a Angular. Ésto se produjo porque, inicialmente, AngularJs trabajaba con código Javascript, pero se sustituyó por Typescript.

Figura 2: Logotipo Angular

Entre las ventajas de Angular destacan:

- **Rendimiento**

- Angular convierte las plantillas del desarrollador en código optimizado para las máquinas virtuales de JavaScript.

- Es universal, ejecuta la vista principal de la aplicación sobre servidores como Nodejs, PHP o .Net para procesarlo y obtener únicamente Html y Css.

- Las aplicaciones Angular cargan únicamente el código necesario para mostrar una vista. No cargan la aplicación completa. De esta manera no se desperdicia tiempo y esfuerzo computacional en código que, a priori, no se necesita.

- **Productividad**

- Angular posee una de las mejores herramientas de comandos de los frameworks web. Angular CLI, permite generar el esqueleto de aplicaciones completas, módulos, componentes y otros muchos elementos. Esto ahorra un tiempo notable al desarrollador, ya que evita que tenga que crear cada uno de los ficheros a mano.
Además de esto, uno de los puntos fuertes de esta herramienta es que permite la previsualización de la aplicación según se va desarrollando.

- Las aplicaciones Angular están muy extendidas, esto hace que los desarrolladores de IDES(Integrated Development Environment) generen extensiones específicas para mejorar la vida del desarrollador. Estas extensiones pueden tener funciones muy variadas, desde generación automática de código hasta la comprobación de código en lo referente a estilo y estructuración del mismo según los estándares establecidos.

- **Testing**

- Por defecto, Angular utiliza Karma para las pruebas unitarias y, para pruebas end-to-end emplea Protractor

2.1.1.2 React

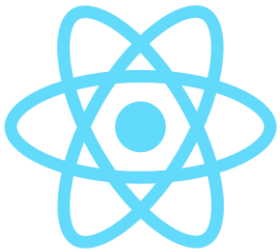

Figura 3: Logotipo React

React es un framework web (aunque también considerado como una librería javascript) que se centra en el desarrollo de interfaces de usuario en aplicaciones SPA.

Fue publicado en 2013 y, hoy en día, es mantenido por desarrolladores independientes, Instagram y Facebook.

La finalidad principal de este framework es facilitar el desarrollo de aplicaciones que utilizan datos que se son modificados constantemente. Todo esto sin perder la facilidad de uso, es declarativo y fácil de combinar con otras tecnologías.

Una peculiaridad de React es que está diseñado únicamente para desarrollar la interfaz de usuario, por lo que se puede combinar con otros frameworks como puede ser Angular.

Como se ha comentado anteriormente, React está enfocado a la creación de interfaces de usuario, por ello puede ser considerado como una librería Javascript, pero se puede considerar framework porque, para lo referente a las partes no gráficas de la aplicación web, posee las extensiones React-Based o Redux. Ahondando así en todas y cada una de las fases de desarrollo de las aplicaciones web.

- **Rendimiento**

La diferencia entre React y otros frameworks como Angular o Ember, es la existencia de un DOM (Modelo de Objeto de Documento) Virtual. Esto quiere decir que, cada vez que se actualiza una vista, lo que cambia y se actualiza es el DOM virtual, que está almacenado en memoria en vez de en el explorador. Esto hace que sea mucho más rápido.

- **Productividad**

- Hay un aspecto de React que mejora mucho un aspecto importante en las páginas web hoy en día, y es el posicionamiento web. Las aplicaciones Javascript y frameworks suelen recibir datos del servidor sin procesar, en formato JSON, que luego tienen que procesar e insertar en el Html para éste ser mostrado por el explorador. Esto es muy negativo de cara al posicionamiento web, ya que el cuerpo de la página no tiene contenido inicialmente.
Gracias al isomorfismo de React, un código Html se renderiza tanto en cliente como el servidor. De esta manera, el trabajo de posicionamiento se simplifica significativamente.

- Al igual que ocurre con Angular, existe una gran variedad de extensiones para IDEs dirigidas a mejorar la experiencia de desarrollo en React.

- **Testing**

Para pruebas unitarias se puede utilizar Karma debido a su compatibilidad con webpack. Otra alternativa es la de utilizar AB Tasty, que recientemente se ha actualizado para ser compatible con todos los nuevos frameworks javascript.

2.1.1.3 Vue

Como el resto de frameworks estudiados anteriormente, Vue o Vue.js es un framework Javascript de código abierto para la creación de interfaces de usuario. Fue publicado en 2014 por Evan You, quién trabajó en empresas tecnológicas como Google y con una amplia experiencia en frameworks web como AngularJS.

Figura 4: Logotipo Vue

A diferencia de otros frameworks, Vue está diseñado desde el inicio para ser adoptado incrementalmente. Es muy sencillo de combinar con otros proyectos o bibliotecas existentes.

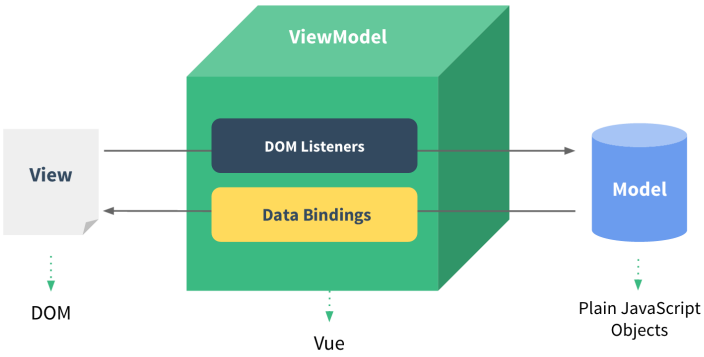

Figura 5: MVC Vue

Como se puede apreciar en la imagen superior, Vue es el intermediario entre la vista y el modelo que se encarga de conservar la sincronía entre los datos. Esto se denomina Data Driven View. Es decir, si los datos cambian, la vista también.

Por todo esto, se puede afirmar que una de las características principales de Vue es que es totalmente reactivo.

- **Rendimiento**

- Una de las grandes ventajas de Vue es el tamaño de su core(en Kbits). Esto hace que sea un código muy compacto y, por tanto, los tiempos de carga y velocidad serán reducidos.
- Al igual que en React, Vue está dotado de un DOM Virtual que aligera la ejecución de las aplicaciones.

- **Productividad**

- Los templates en Vue son escritos en Html directamente y pueden ser modificados fácilmente por cualquier miembro del equipo de desarrollo.
- Así como Angular tiene Angular-Cli como herramienta de generación de estructuras básicas (scaffolding), Vue tiene Vie-Cli. Eso hace que el desarrollo sea más rápido.

- **Testing**

Desde la página oficial de Vue, se recomienda el uso de Karma para el testing unitario.

2.1.1.4. Bootstrap

Bootstrap es un framework de código abierto para diseño web lanzado en 2011 por Twitter.

Este framework se centra únicamente en la interfaz de usuario. Esto hace que se utilice normalmente en conjunto con otros frameworks como Angular.

Figura 6: Logotipo Bootstrap

Es uno de los frameworks más destacados y extendidos de todos los frameworks front-end. Esto es debido a que ofrece la posibilidad de crear un sitio web totalmente responsive mediante el uso de librerías CSS, es decir, adapta la vista al tamaño del dispositivo. Además, ofrece un gran abanico de elementos ya desarrollados listos para ser utilizados como botones, menús, cuadros e incluso diferentes fuentes de letra.

Como el resto de frameworks mencionados en esta sección, Bootstrap tiene una gran comunidad de desarrolladores que ayudan a la evolución y mejora del mismo.

- **Rendimiento**

Uno de los puntos débiles de este framework es el rendimiento en algunos casos. El problema es que, como ya se ha comentado anteriormente, Bootstrap provee al usuario de una gran cantidad de elementos visuales y, aunque solo vayamos a utilizar una pequeña fracción de ellos, hay que cargar todos y cada uno de los elementos. Esto hace que la carga de la vista se ralentice.

- **Productividad**

A pesar de todas las posibilidades que ofrece este framework, su nivel de complejidad es bajo. Esto hace que un desarrollador con poca experiencia pueda generar una aplicación totalmente responsive sin mucho esfuerzo.

- **Testing**

Al ser un framework destinado únicamente a elementos CSS, no se realizan pruebas como en los demás. Existen herramientas de test que comprueban el aspecto responsive de la aplicación y así asegurar que la página se está mostrando correctamente en todos los dispositivos.

Elección

Tras realizar el estudio de los posibles frameworks, se ha optado por Angular y Bootstrap.

Se ha optado por Angular por su gran inclusión en el mundo empresarial, por el amplio abanico de librerías que posee y por una cuestión personal, ya que ya estaba familiarizado con el framework tras estar trabajando con él en las prácticas en empresa durante 8 meses.

En cuanto a Bootstrap, es un framework que hoy en día está presente en la mayoría de aplicaciones web, ya que facilita el desarrollo de la UI y, además, transforma la aplicación en una aplicación responsive.

2.1.2. Productos de Almacenamiento Cloud

En el estudio de tecnologías se estudiaron también las principales opciones de almacenamiento en la nube. Esto se hizo debido a que será necesario almacenar las escalas de evaluación en algún lugar de fácil acceso para cada uno de los usuarios de la aplicación.

Una de las ventajas de esta modalidad de almacenamiento es que permite a las empresas o desarrolladores, prescindir de hardware específico. Contratando estos servicios, se paga según el tráfico que reciban sus máquinas y, además, permiten un sistema escalable. Si se necesita más espacio o más capacidad de procesamiento, se le otorga automáticamente. Esta investigación se centrará en el aspecto de almacenamiento de estos servicios, ya que también proveen servicios de procesamiento.

Tras investigar cuáles eran las herramientas que más se están empleando en el mundo empresarial, se centra la investigación en dos, Amazon Web Services y Google Cloud Platform. Cabe destacar también Azure, pero se ha omitido por no estar familiarizado con él.

2.1.2.1. Amazon Web Services vs Google Cloud Platform

Figura 7: Amazon vs Google

En cuanto a almacenamiento se refiere, AWS, ofrece varias posibilidades según las necesidades y las características de la información que se vaya a tratar.

Amazon Simple Storage Service (S3) y Cloud Storage Standard, para el almacenamiento de objetos, que permite el fácil análisis de información. El precio de los servicios de Google es ligeramente menor en este caso.

Amazon Elastic Block Store (Amazon EBS) y Google Persistent Disk, diseñados para aplicaciones con necesidades de fácil y rápido acceso a información, con control de acceso, alta disponibilidad y datos sensibles a ataques. En cuanto al coste de estos servicios, no se puede establecer un precio determinado ya que depende de las peculiaridades de la aplicación.

Amazon Elastic File System (EFS) y Google Cloud Storage Nearline, destinados a albergar información a la que no se va a acceder frecuentemente. La latencia de conexión es mayor que en el resto de los servicios descritos y, por ello, el coste del servicio es menor. Para este tipo de servicios, Amazon ofrece un precio más competitivo.

Para decidir entre ambos, se debe realizar una comparativa de precios y contratar aquellos que mejor se adapten a las necesidades de la aplicación.

2.1.3. Librerías Angular para la recogida dinámica de datos

Para el proceso de creación de escalas de evaluación, hubo que realizar un amplio estudio de todas las librerías disponibles para Angular que permitieran la recogida dinámica de datos. La recogida de datos debía ser dinámica debido a que las escalas de evaluación varían en el número de tareas, por lo que no se puede establecer un número fijo de ellas.

El mayor problema que se encontró en este apartado es que no solo se necesitaba recogida dinámica de datos en las tareas, si no también en el número de grados de evaluación de cada una de las tareas.

A continuación, se describen las opciones que se barajaron para la implementación.

2.1.3.1. Ng2-smart-table

Librería creada por el equipo de Akveo. Se encuentra disponible como un paquete npm, por lo que su instalación es muy simple, únicamente tenemos que ejecutar el siguiente comando situándonos con la consola en la raíz del proyecto:

```
npm install --save ng2-smart-table
```

Proporciona una interfaz intuitiva, con posibilidades de filtrado, y con capacidad para cargar datos recogidos a partir de la api dispuesta por el servidor. Éste último es uno de sus puntos fuertes, ya que resulta sencillo de implementar y su ejecución es rápida.

2.1.3.2. Ng-editable-table

Librería creada por el desarrollador BennyFranco. Como la anterior, se encuentra disponible como un paquete npm, por lo que su instalación es muy simple, únicamente tenemos que ejecutar el siguiente comando situándonos con la consola en la raíz del proyecto:

```
npm install ng-editable-table --save
```

Proporciona una de las interfaces más cuidadas de las opciones barajadas. Permite únicamente añadir, editar y eliminar filas de la tabla.
En cuanto a la recogida y representación de datos de servidor, no dispone de un mecanismo establecido, por lo que el proceso podría ser complejo.

Elección

Tras realizar el análisis anterior, se decidió que la librería que mejor encajaba en el proyecto, era **Ng2-smart-table.**

Se tomó esta decisión debido a que la obtención y procesado de información desde servidor era vital a la hora de editar una escala de evaluación. Además, su integración en el proyecto era más sencilla que en el caso de **ng-editable-table**, que obliga a crear módulos extra.

2.2 Estudio de Escalas de Evaluación

En este apartado se ha realizado un estudio sobre los métodos de evaluación empleados por los profesionales sanitarios.

Para poder implementar correctamente el proceso de evaluación y creación de escalas de valoración, se debe tener claro de qué elementos se compone una escala y cómo obtener el resultado de ésta.

Se comenzó el estudio contactando con profesionales del sector, que expusieron algunas de las escalas de evaluación que emplean en su día a día. Una vez recogida la información necesaria, se comenzó el estudio de la morfología de las escalas.

Como punto de partida, se observó que las escalas de evaluación se componen de dos secciones. Una primera donde se detallan las tareas y el grado con el que el paciente las supera, y un puntaje asignado a cada grado y, por último, una segunda parte, donde según el intervalo en el que se encuadre el puntaje final, se obtendrá el resultado.

A continuación, se muestra un ejemplo de una escala de evaluación. Se trata de la escala Barthel, empleada por los especialistas sanitarios para la valoración funcional de un paciente y así medir su grado de dependencia a lo largo de la rehabilitación.

Índice Barthel		
Actividad	Descripción	Puntaje
Comer	1. Incapaz 2. Necesita ayuda para cortar, extender mantequilla, usar condimentos, etc. 3. Independiente (la comida está al alcance de la mano)	0 5 10
Trasladarse entre la silla y la cama	1. Incapaz, no se mantiene sentado 2. Necesita ayuda importante (1 persona entrenada o 2 personas), puede estar sentado 3. Necesita algo de ayuda (una pequeña ayuda física o ayuda verbal) 4. Independiente	0 5 10 15
Aseo personal	1. Necesita ayuda con el aseo personal 2. Independiente para lavarse la cara, las manos y los dientes, peinarse y afeitarse	0 5
Uso del retrete	1. Dependiente 2. Necesita alguna ayuda, pero puede hacer algo solo 3. Independiente (entrar y salir, limpiarse y vestirse)	0 5 10
Bañarse o Ducharse	1. Dependiente 2. Independiente para bañarse o ducharse	0 5
Desplazarse	1. Inmóvil 2. Independiente en silla de ruedas en 50 m 3. Anda con pequeña ayuda de una persona (física o verbal) 4. Independiente al menos 50 m, con cualquier tipo de muleta, excepto andador	0 5 10 15
Subir y bajar escaleras	1. Incapaz 2. Necesita ayuda física o verbal, puede llevar cualquier tipo de muleta 3. Independiente para subir y bajar	0 5 10
Vestirse y desvestirse	1. Dependiente 2. Necesita ayuda, pero puede hacer la mitad aproximadamente, sin ayuda 3. Independiente, incluyendo botones, cremalleras, cordones, etc.	0 5 10
Control de heces	1. Incontinente (o necesita que le suministren enema) 2. Accidente excepcional (uno/semana) 3. Continente	0 5 10
Control de orina	1. Incontinente, o sondado incapaz de cambiarse la bolsa 2. Accidente excepcional (máximo uno/24 horas) 3. Continente, durante al menos 7 días	0 5 10

Figura 8: Escala Barthel

Puntaje	Clasificación
<20	Dependencia total
20 – 35	Dependencia severa
40 – 55	Dependencia moderada
60 – 95	Dependencia leve
100	Independencia

Figura 9: Escala Barthel Resultados

3 Diseño

A continuación, se detallará el diseño definido para crear la aplicación.

Hay ciertos aspectos sobre la arquitectura que no se podrán definir, como es la Base de Datos del centro médico. Esto es debido a que no se ha tenido acceso a la misma.

3.1 Descripción de la Arquitectura

Para la arquitectura de la aplicación se ha diseñado un sistema compuesto por una aplicación Angular, formada, a su vez, a partir de código Html, Css y Typescript, que se ejecutará sobre un servidor NodeJS y este se conectará con la base de datos del centro médico o al servidor de almacenamiento en la nube.

En la base de datos del centro médico se almacenarán los resultados de las evaluaciones y, en la nube, se guardarán las escalas de evaluación vacías para poder compartirlas con el resto de los usuarios de la aplicación.

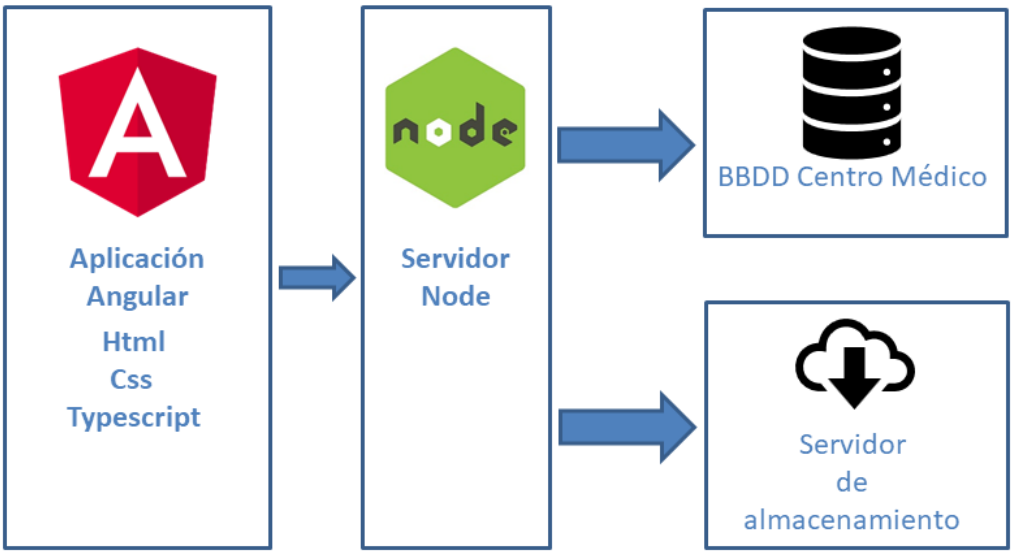

Figura 10: Arquitectura de la Aplicación

Tanto la Base de Datos como el Servidor de almacenamiento son elementos que no se han podido probar, ya que se han considerado para iteraciones futuras del software.

3.2 Análisis de Requisitos

En este apartado se detallará el análisis de requisitos realizado.

Los requisitos son aquellas condiciones o capacidades que debe cumplir un sistema para cumplir con los objetivos y expectativas de las partes interesadas en el proyecto.

Dividimos los requisitos en dos tipos. Los requisitos funcionales y los no funcionales. Los funcionales definen acciones que el sistema ha de cumplir al ejecutarse, mientras que los no funcionales definen características de funcionamiento, como por ejemplo el lenguaje del sistema o el número de usuarios.

A continuación, se enumeran los requisitos definidos para este proyecto.

Requisitos Funcionales		
Usuario		
ID	**Acción**	**Descripción**
1º Iteración		
Rf-1	Iniciar Sesión (Log-in)	El usuario debe poder iniciar sesión introduciendo su usuario y contraseña
RF-2	Cerrar Sesión (Log-out)	El usuario debe poder cerrar sesión pulsando en un botón definido.
RF-3	Crear Escala	El usuario podrá crear una escala de evaluación con un número variable de tareas.
RF-4	Evaluar	El usuario podrá evaluar un paciente cargando la escala deseada.
RF-5	Listar Escalas	El usuario podrá obtener la lista de todas las escalas definidas en el sistema.
RF-6	Guardar Diagnóstico	El usuario podrá guardar, en la base de datos, el diagnóstico obtenido a través de una escala de evaluación.
2º Iteración		
RF-7	Compartir Escala	El usuario podrá compartir una escala con el resto de los usuarios de la aplicación.
RF-8	Administrar Escalas	El usuario podrá gestionar sus escalas editándolas o eliminándolas.
Administrador		
RF-9	Administrar Usuarios	El Administrador podrá añadir, eliminar y editar propiedades de usuarios

Tabla 1: Requisitos Funcionales

Requisitos No Funcionales		
ID	**Acción**	**Descripción**
RNF-1	Modificación de datos de acceso	Los datos de acceso a la aplicación solo podrán ser modificados por el administrador.
RNF-2	Encriptación de datos	Los resultados de las evaluaciones deben encriptarse mediante el algoritmo RSA
RNF-3	Diseño Responsive	La aplicación debe tener un diseño Responsive a fin de asegurar la correcta visualización en múltiples dispositivos.
RNF-4	Interfaz para navegadores web	La aplicación deberá ser implementada para navegadores web con Html5 y Javascript
RNF-5	Sistema de datos médicos	La aplicación deberá cumplir con la ley de protección de datos médicos

Tabla 2: Requisitos No Funcionales

El desarrollo de este proyecto ha sido iterativo, por lo que hay ciertos requisitos que no se han implementado en esta primera iteación.

Estos requisitos que no se han implementado son: RF-7 Compartir Escala, RF-8 Administrar Escalas y RF-9 Administrar Usuarios.

3.3 Diagramas de casos de uso

La figura muestra el diagrama de casos de uso de la aplicación. Se puede apreciar la existencia de dos roles, el de usuario, correspondiente al profesional sanitario, y el de administrador, que se encargará de administrar los usuarios.

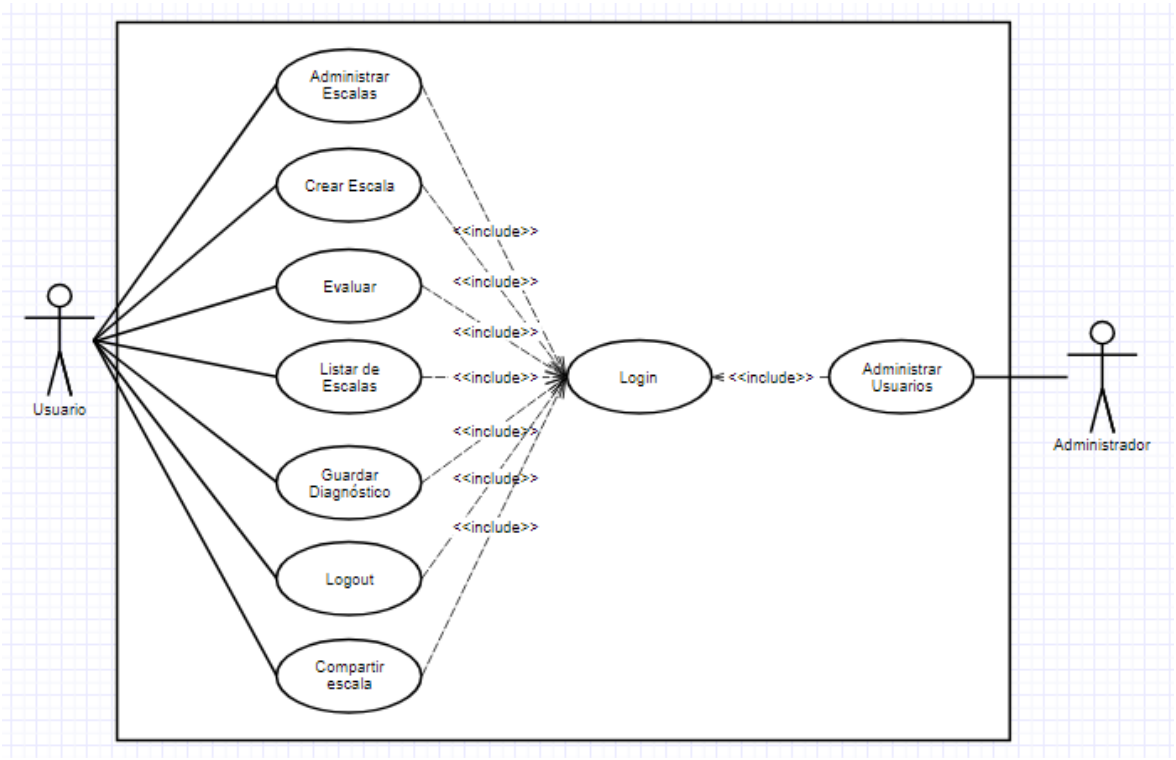

Figura 11: Diagrama de Casos de Uso de la Aplicación

3.4 Diagrama de Clases

Para el diagrama de clases se ha diseñado un diagrama sobre el modelo de datos en conjunto con las clases empleadas en el desarrollo. De esta manera se obtiene un diagrama más completo.

En cuanto al modelo de datos empleado, se compone de las siguientes clases:

- ***Escala***: Representa las escalas de evaluación que gestionará la aplicación.
    - ***Tarea***: Cada una de las tareas que componen una escala de evaluación.
        - ***Opción***: opciones que componen las tareas.
    - ***Resolución***: Resultado de la escala de evaluación.
        - ***Posibilidades***: cada una de las posibles soluciones que proporciona la escala de evaluación según el puntaje obtenido.
- ***Diagnóstico***: Contendrá el resultado de la escala de evaluación.
- ***Usuario***: Representa al profesional sanitario que utilizará la aplicación.

Para la implementación se han declarado una serie de clases Typescript que albergarán la funcionalidad de la aplicación. Se hablará solo de las que implementen funciones con importancia, ya que cada componente Angular, es una clase Typescript. Esto da lugar a clases para elementos visuales como pueden ser la cabecera o la barra de navegación.
A continuación, se detallarán las clases empleadas:

- ***CreateScaleComponent***: Clase que implementa el proceso de creación de una escala de evaluación.
- ***ListScalesComponent:*** Se encarga de listar todas las escalas de evaluación disponibles para el usuario que ha hecho login.
- ***EvaluateComponent:*** Clase encargada del proceso de evaluación de un paciente a través de una escala de evaluación.

En la figura posterior, Figura 11, se puede apreciar que las entidades que representan el modelo de datos carecen de métodos ya que son una representación de datos, no una clase al uso.

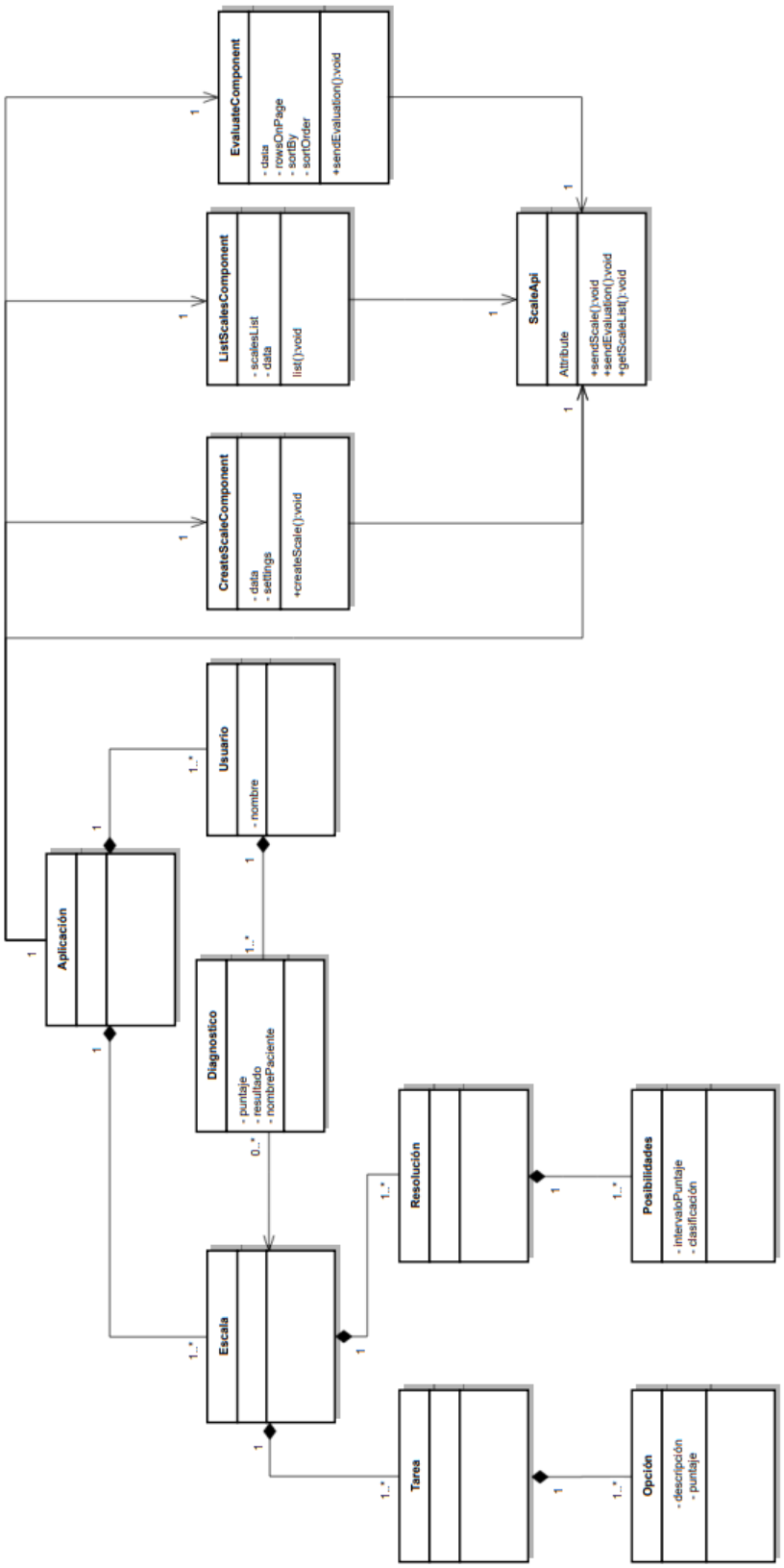

Figura 12: Diagrama de Clases de la aplicación

4 Desarrollo

En este apartado se describe todo el trabajo de desarrollo llevado a cabo en la primera iteración del proyecto. Hay ciertos requisitos que se han dejado para posteriores iteraciones.

4.1 Herramientas

4.1.1 Hardware

La implementación de la aplicación se ha desarrollado en un ordenador portátil con un procesador i7 y 16gb de memoria RAM.
No ha sido necesario el uso de ningún otro hardware especial.

4.1.1 Software

Se ha empleado una gran variedad de herramientas software para el desarrollo de la aplicación. A continuación, se detallará cada una de ellas.

4.1.1.1 Windows 10

Sistema operativo Windows 10 sobre el que se han ejecutado todas las demás herramientas.
Se decidió emplear Windows en vez de Linux por una cuestión personal, ya que siempre he desarrollado este tipo de aplicaciones en Windows.

4.1.1.2 Visual Studio Code

Editor de texto de código abierto creado y mantenido por Microsoft. Su uso está muy extendido en el mundo del desarrollo web debido a la gran cantidad de complementos disponibles que hacen el desarrollo más rápido y cómodo.

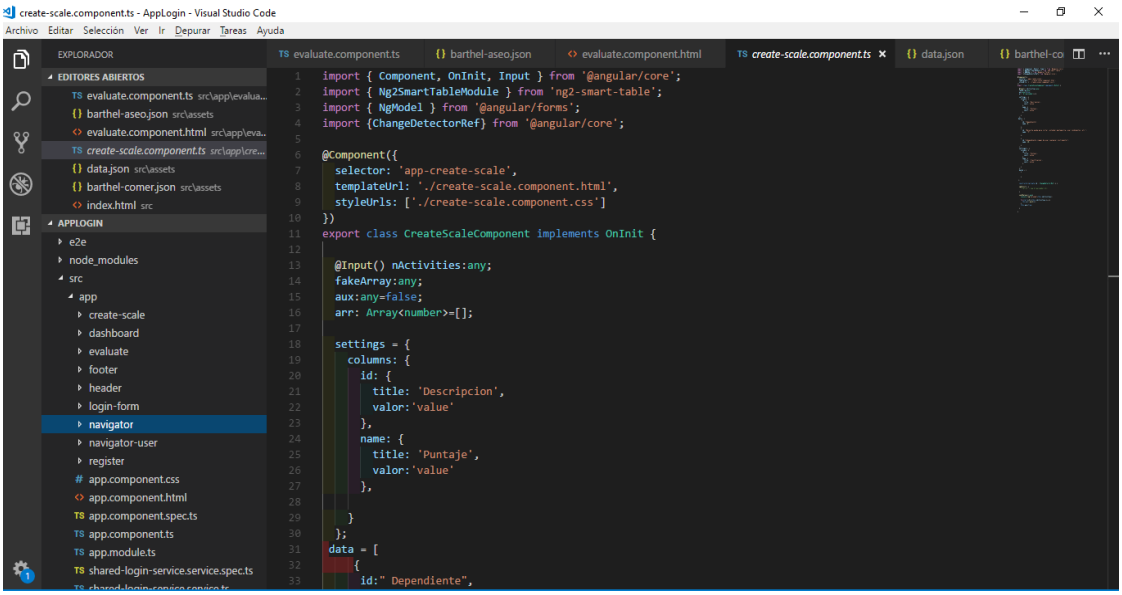

Figura 13: Visual Studio Code

4.1.1.3 Angular 4

Se ha empleado la versión 4 del framework debido a que, cuando se comenzó el desarrollo, era la versión disponible.

El comienzo con este framework fue sencillo, ya que había trabajado con él en las prácticas en empresa. El primer paso fue instalarlo en el ordenador personal donde se iba a desarrollar la aplicación.

Para la instalación del framework, comenzamos instalando la última versión del servidor NodeJS. Además de hacer posible la ejecución de aplicaciones, NodeJS proporciona un ecosistema de paquetes, NPM, con el número de librerías de código abierto más grande del mundo.

Una vez instalado NodeJS, pasamos a instalar el propio framework. Esto se puede hacer a través de NPM:

```
npm install -g @angular/cli
```

Angular Cli es la herramienta de línea de comandos de Angular. Esta herramienta es casi imprescindible a la hora de desarrollar una aplicación en Angular ya que facilita el trabajo de desarrollo notablemente.

Una vez instalado Angular Cli, ejecutamos el siguiente comando para crear la aplicación:

```
ng new ApplicationName
```

Solo con esto, ya se ha creado el esqueleto completo de la aplicación, incluyendo un hola mundo para que el desarrollador compruebe que todo está funcionando correctamente.

Para ejecutar la aplicación basta con acceder con la consola al directorio de la aplicación y ejecutar:

```
ng serve
```

Finalizada la instalación, comenzamos con el desarrollo de la aplicación. Para ello, se generan los componentes que van a formar la aplicación.

Cada componente consta de cuatro archivos:

- **Archivo Html**

HyperText Markup Language, emplea el lenguaje marcado para el desarrollo de páginas web. [22]

En el archivo HTML del componente, se definirá la vista del componente. En el proyecto desarrollado, el código HTML se combinará con el código del framework **Bootstrap.**

```
<div class="collapse navbar-collapse " id="navbarColor03">
  <ul class="navbar-nav mr-auto">
    <li class="nav-item active ml-4">
    <a class="nav-link font-weight-bold text-primary " href="#">Home <span class="sr-only">(current)</span></a>
    </li>
    <li class="nav-item ml-2">
    <a class="nav-link font-weight-bold" routerLink="createScale">Crea tu Escala</a>
    </li>
    <li class="nav-item ml-2 ">
    <a class="nav-link font-weight-bold" routerLink="evaluate">Evalúa</a>
    </li>
    <li class="nav-item ml-2">
    <a class="nav-link font-weight-bold" href="#">Lista de Escalas</a>
    </li>
  </ul>
  <ul class="navbar-nav  pull-right">
    <li class="nav-item ml-2">
      <a class="nav-link font-weight-bold " href="#">
        <i class="fa fa-twitter fa-2x text-primary " aria-hidden="true"></i>
      </a>
    </li>
    <li class="nav-item ml-2">
      <a class="nav-link font-weight-bold" href="#">
        <i class="fa fa-facebook-square fa-2x text-primary " aria-hidden="true"></i>
      </a>
    </li>

  </ul>
```

Figura 14: Html y Bootstrap

En el ejemplo, se puede apreciar el código Html combinado con el código Bootstrap. Se aprecia el código Bootstrap observando los fragmentos de código *class*. En este caso se está haciendo uso de la barra de navegación *navbar-nav* del framework y se está definiendo, a través de *mr-auto*, que la barra de navegación tenga los márgenes automáticos, es decir, que se adapten al *div* que la contenga.

- **Css**

Las siglas Css significan *Hojas de estilo en cascada*. Se trata de un lenguaje de diseño gráfico que emplea un lenguaje marcado para el diseño de los elementos visuales que declarados en las vistas Html.

Para este proyecto se empleó una hoja de estilos descargada de un sitio web llamado *Bootswatch.com* que ofrece temas Css gratuitos.
De entre toda la oferta, se escogió el tema *Materia*, que ofrece un diseño visual fresco y sencillo que encaja a la perfección con la apariencia que se quería para la aplicación.

```
.btn-primary:hover,
.btn-primary:active:hover {
    background-color: #0d87e9;
}

.btn-primary:active {
    -webkit-box-shadow: 2px 2px 4px rgba(0, 0, 0, 0.4);
    box-shadow: 2px 2px 4px rgba(0, 0, 0, 0.4);
}

.btn-primary:after {
    content: "";
    display: block;
    position: absolute;
    width: 100%;
    height: 100%;
    top: 0;
    left: 0;
    margin-left: 0;
    background-image: radial-gradient(circle, #fff 10%, transparent 10.01%);
    background-repeat: no-repeat;
    background-size: 1000% 1000%;
    background-position: 50%;
    border: none;
    opacity: 0;
    pointer-events: none;
    -webkit-transition: background .5s, opacity 1s;
    transition: background .5s, opacity 1s;
}
```

Figura 15: Css

Este es un fragmento de la hoja Css empleada en el proyecto. Se puede ver que modela los botones ofrecidos por Bootstrap para adaptarlos a las necesidades del proyecto. En concreto, se muestra cómo define las propiedades para un botón con un link activo, para el aspecto del botón una vez pulsado y, también, para el aspecto que debe mostrar el botón al pasar el cursor por encima.

Aunque se haya descargado este tema Css, se han aplicado cambios para obtener el aspecto deseado para a aplicación desarrollada en este proyecto.

- **Typescript**

Como se expuso en el Capítulo de Estudio de Tecnologías, Angular 4 ya no trabaja con código Javascript, sino que hace uso de código Typescript para la codificación de la parte controladora del módulo. [23]

Aunque no se utilice código JavaScript, al ser Typescript una extensión de Javascript, cualquier elemento de este, funcionará correctamente.

```typescript
import { Component, OnInit, Input } from '@angular/core';
import { Ng2SmartTableModule } from 'ng2-smart-table';
import { NgModel } from '@angular/forms';
import {ChangeDetectorRef} from '@angular/core';

@Component({
  selector: 'app-create-scale',
  templateUrl: './create-scale.component.html',
  styleUrls: ['./create-scale.component.css']
})
export class CreateScaleComponent implements OnInit {

  @Input() nActivities:any;
  fakeArray:any;
  aux:any=false;
  arr: Array<number>=[];

  settings = {
    columns: {
      id: {
        title: 'Descripcion',
        valor:'value'
      },
      name: {
        title: 'Puntaje',
        valor:'value'
      },

    }
  };
```

Figura 16: Typescript

En esta figura, se muestra parte del código Typescript del componente *CreateScale*.

Se pueden apreciar varias secciones. Una primera donde se especifican otros módulos que se emplearán en el componente *CreateScale*. Una segunda sección, *@component*, propia de Angular, donde se definen ciertos aspectos del componente como la forma de referirnos a él, *selector*, y poder insertarlo en otros módulos, *templateUrl* donde se especifica la vista Html que empleará y *styleUrls*, para localizar la hoja de estilos con la que se va a trabajar. Por último, la sección de declaración de la clase, que albergará la funcionalidad del componente.

Además del controlador del componente, Angular crea otro archivo Typescript para facilitar el testing del mismo.

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';

import { NavigatorComponent } from './navigator.component';

describe('NavigatorComponent', () => {
  let component: NavigatorComponent;
  let fixture: ComponentFixture<NavigatorComponent>;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [ NavigatorComponent ]
    })
    .compileComponents();
  }));

  beforeEach(() => {
    fixture = TestBed.createComponent(NavigatorComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

Figura 17: Código Tester Typescript

En esta figura se aprecia el esqueleto básico que crea la herramienta Angular Cli. Se puede observar que crea una instancia del componente para realizar pruebas específicas.

4.1.1.4. Bootstrap

Framework empleado para implementar las vistas de la aplicación de una forma sencilla y que la aplicación fuera responsive, es decir, las vistas se adaptaran a los dispositivos que las estuvieran mostrando.

Al igual que con el framework anterior, ya estaba familiarizado con él y tan solo tuve que profundizar en algunos aspectos como la definición de tamaños de elementos.

4.2. Proceso de Desarrollo

Una vez preparado todo el entorno de desarrollo, se comenzó a desarrollar en función de lo definido en la etapa de Análisis y Diseño.

En cuanto al alcance de la implementación, hay ciertos objetivos que no se pudieron realizar debido a falta de tiempo y a falta de medios, ya que no se tiene acceso al sistema de ningún centro médico. Por ejemplo, la parte servidora, no se ha podido implementar.

Se muestra a continuación el trabajo de desarrollo elaborado en esta primera iteración de la implementación de la aplicación.

4.2.1. Implementación de Componentes Básicos

Se comienza el desarrollo con la implementación de los componentes básicos. Estos son:

- **Componente de Login: Login-form**

  Componente que implementa el requisito RF-1.

  Se genera un componente para el proceso de Login. Esto se realiza mediante la herramienta de comandos de angular, Angular Cli, que genera los archivos necesarios.

  Se codifica primero la vista del componente en el archivo Html. Éste se compone de un formulario que solicita el usuario y la contraseña. Para el formulario se hizo uso de Bootstrap y de la hoja de estilos descargada.

  Una vez implementada la vista, se pasa al controlador. Este proceso debería hacerse implementando un servicio que contactara con el sistema del centro médico y que éste, le autorizara el acceso, pero como no se tiene acceso a ninguno, se ha implementado un proceso de autenticación sencillo basado en una comparación de los datos introducidos por el usuario y los almacenados en local por la aplicación.

- **Componente Home: Dashboard**

  Se crea un nuevo componente al que se accederá una vez el usuario se haya autentificado.

  Para este componente, se crea una vista atractiva con imágenes y textos que exponen las posibilidades que ofrece la aplicación. No es necesario implementar métodos ni servicios ya que Angular permite establecer links a otros componentes en los propios botones y elementos Html.

- **Componente Navegador: Navigator**

  Este componente será el encargado de la navegación entre los distintos componentes de la aplicación.

  Se escoge una barra de navegación horizontal del tema Css escogido, que empleará además Bootstrap para adaptarse a las dimensiones del explorador.

- **Componentes Encabezado y Pie de Página: Header y Footer**

  Se generan estos componentes con la única funcionalidad de establecer un marco visual atractivo.

Solo se desarrolla la vista del componente empleando el tema Css y algún elemento Bootstrap.

Esta etapa de desarrollo es sencilla. El único aspecto que requirió algo de tiempo e investigación, fue el ocultar la barra de navegación cuando el usuario no está autentificado. Angular provee aplicaciones SPA formadas por componentes. Esto añade una complejidad extra en este tipo de procesos.

Para que aparecieran los componentes adecuados al acceder a la aplicación, se creó un servicio *sharedservice*, que comparte información entre varios componentes, y en conjunto con la sentencia *NgIf* de Angular, se consiguió ocultar la barra de navegación al inicio de la ejecución, y una vez autentificado, se mostrará.

4.2.2. Implementación del componente de Creación de Escalas

Este componente representa el requisito RF-3.

La implementación de este componente sea, probablemente, la más compleja.
Tras realizar el trabajo de investigación en lo referente a la recogida dinámica de datos, se comienza el desarrollo.

Una vez diseñada la estructura de la vista, se integra la librería ng2-smart-table. Haciendo uso de la guía proporcionada por el proveedor de la librería, se instala en el proyecto y se declara en varios módulos de la aplicación para conseguir un correcto funcionamiento de ésta. A continuación, se instancia la librería añadiendo la línea de código al Html del componente:

```
    <ng2-smart-table [settings]="settings" [source]="data"></ng2-smart-table>
```

En *settings* se definen las propiedades de la tabla, como el número de columnas y sus nombres, y en *source*, la fuente de datos, tanto de carga como de guardado. Ambas propiedades trabajan con datos en formato tipo JSON.

Aquí surge un problema y es que la tabla que proporciona la librería, no se adapta del todo a las necesidades de nuestra aplicación, que puede ser variable en número de tareas y en número de descripciones y puntajes. Debido al tiempo que se tiene para el desarrollo de trabajo, se decide implementar un sistema por el cual, el usuario ha de introducir el número de tareas o actividades que va a tener la escala.

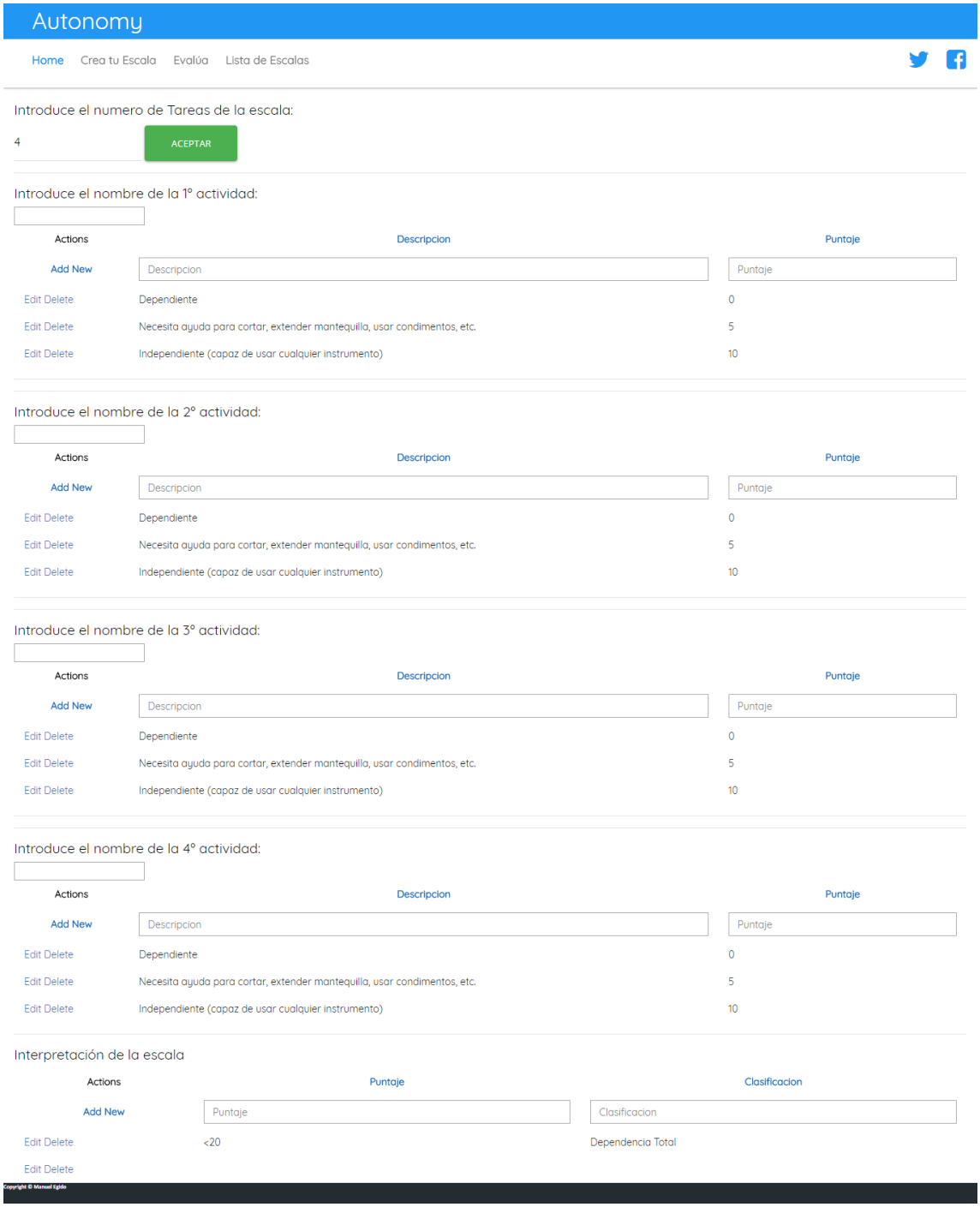

Figura 18: Vista de la creación de Escala de Evaluación de la aplicación

Esta es parte de la vista que se genera al introducir el usuario el número de tareas.
Se le solicita introducir el nombre de las actividades y su descripción.
El usuario introduce 4 tareas y se crean las tablas para cada una de ellas y una tabla final para la interpretación de la escala.

Todos los datos introducidos se recogen y se envían al servidor mediante el servicio sendScale. No se ha podido probar con el servidor por lo explicado anteriormente, pero se enviarían en formato JSON.

4.2.3. Implementación del componente de Evaluación

Este componente representa la implementación del requisito RF-4.

En este componente se implementa la funcionalidad de Evaluar a un paciente.

Al acceder, se mostrará la lista de escalas de evaluación disponibles, y se seleccionará alguna de ellas.

Una vez seleccionada la escala, se cargan los datos de la escala a través de una llamada a un servicio que obtendrá los datos de la escala de evaluación del servidor, *getScaleList*. Estos datos se obtienen en JSON, se procesan, y se guardan en un array. Posteriormente, en la vista y mediante la sentencia Angular *NgFor*, se muestran en una tabla que tendrá, a su vez, campos de tipo *checkbox* para seleccionar la opción deseada.

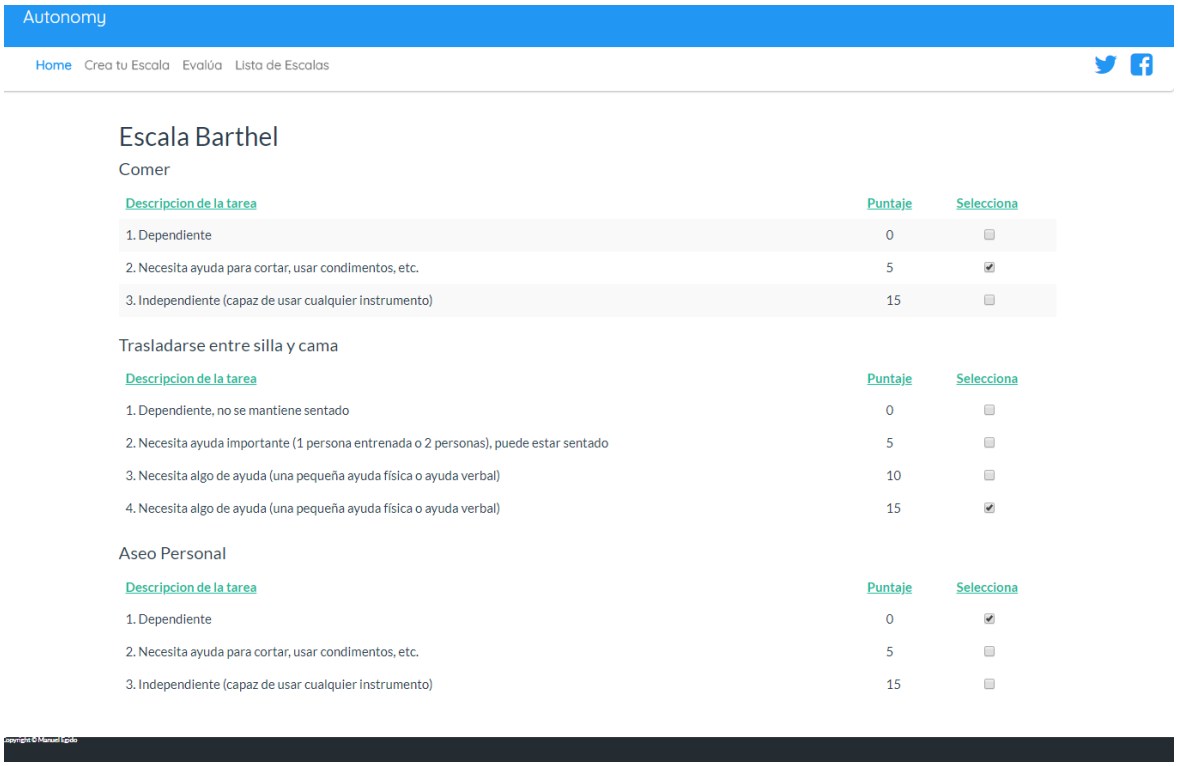

Figura 19: Vista del proceso de evaluación de la aplicación

Una vez cumplimentada la escala, el usuario pulsará el botón de *evaluar* y la aplicación mostrará el resultado junto con la interpretación. Una vez hecho esto, se da la posibilidad de enviar el resultado a la base de datos del centro mediante el servicio *sendEvaluation*, que enviará el resultado en formato JSON al servidor, y este a la BBDD. Este proceso representa el requisito RF-6. En esta iteración del proyecto, se ha realizado el proceso de guardado en local, en un archivo JSON.

4.2.4. Implementación del componente de Lista de Escalas

La implementación de este componente corresponde con el requisito RF-5.

Este componente se conectará con el servidor mediante el servicio *getScaleList*. La llamada a este servicio se implementará en el método ngOnInit de Angular, que permite la llamada automática al servicio cuando se quiere cargar el componente.
De esta manera, cuando se acceda al componente, los datos ya estarán disponibles para ser mostrados.

Los datos se obtienen en formato JSON, se procesan, se guardan en un array y, para mostrarlos, se recorrerá el array mediante la herramienta de Angular ngFor, que irá recorriendo el array mostrando cada elemento de éste.

Trabajo para futuras iteraciones:
Este ha sido el trabajo de desarrollo realizado en esta primera iteración del trabajo. Queda así pendiente el desarrollo en posteriores iteraciones de los requisitos RF-7 Compartir escala y RF-8 Administrar Escalas.

5 Integración, pruebas y resultados

A continuación, se describirán las pruebas realizadas sobre la aplicación para asegurar su correcto funcionamiento.

Como se ha comentado en apartados anteriores, no se ha podido comprobar la funcionalidad completa de la aplicación debido que a que hay elementos con los que interactúa, a los que no se ha podido acceder. Estos elementos son, la Base de Datos de un centro médico y un servidor que permita la conexión con ésta.

Por todo esto, las pruebas se centrarán sobre todo en la interacción del usuario con la aplicación, es decir, pruebas funcionales. Al seguir el desarrollo un modelo iterativo y ser esta entrega la primera iteración, las pruebas no funcionales deberían realizarse al finalizar el proyecto.

5.1 Pruebas Funcionales

Las pruebas funcionales son aquellas que se basan en la ejecución y revisión de las funcionalidades diseñadas previamente para el software.

A continuación, se detallan las pruebas realizadas a la aplicación desarrollada en la primera iteración del proyecto.

5.1.1. Pruebas unitarias

Este tipo de pruebas comprueban la lógica, la funcionalidad y si se cumplen las especificaciones para cada componente de la aplicación.

Se han realizado sobre que cada módulo de la aplicación independientemente de los otros para asegurar que los fallos no son producidos por acarreo de otros módulos y así tenerlos localizados.

A continuación, se detallan las pruebas unitarias realizadas a los módulos principales de la aplicación.

- ***Módulo CreateScale - Creación de Escala***

Prueba Unitaria U-1: Comprobación de modificación de archivo Json a través de tablas dinámicas.

Descripción: Se introducen datos en las tablas para comprobar que se modifica el archivo Json que representa la escala.
Entrada: Se crea una nueva fila con una descripción y un puntaje en alguna tarea ya creada.
Salida: Archivo Json de la escala se modifica apareciendo el campo de la nueva fila.

- ***Módulo Evaluate - Evalúa***

Prueba Unitaria U-2: Comprobación de recogida de datos a través de formulario de escala.

Descripción: Se rellena el formulario de una escala y se comprueba que los resultados se almacenen correctamente en la variable puntaje.
Entrada: Se carga una escala y se pulsa en varios campos.
Salida: La variable puntaje refleja la suma de los valores seleccionados.

- ***Módulo Lista de escalas - ListScales***

Prueba Unitaria-3: Comprobación de procesamiento de datos de archivo Json a array Typescript

Descripción: Se comprueba el procesado de datos que representan la lista de escalas y su copia a un array en Typescript.
Entrada: Se lee de un archivo Json y se procesa hasta insertarlo en un array en lenguaje Typescript.
Salida: Impresión de array que contiene la lista de escalas reflejada en el archivo Json.

5.1.2. Pruebas de integración

Las pruebas de integración se realizan para asegurar el correcto funcionamiento de los componentes trabajando en conjunto. Se realizan después de las pruebas unitarias.

En esta iteración del proyecto no ha sido necesario implementarlas ya que los módulos funcionan independientemente.

5.1.3. Pruebas de validación

Las pruebas de validación comprueban que las funcionalidades reflejadas en el análisis de requisitos se ejecutan correctamente.

Este tipo de pruebas se pueden clasificar en dos grupos dependiendo de quién las realice. Si las realiza el cliente en el entorno de desarrollo, se denominan pruebas Alfa. Si las pruebas se realizan en el entorno del cliente sin la presencia del desarrollador, se califican como pruebas Beta.

Para estas pruebas se ha contado con la colaboración de una profesional del ámbito sanitario con experiencia, en concreto con una Terapeuta Ocupacional. Las pruebas se han realizado en el entorno de desarrollo, por lo que se consideran pruebas Alfa.

A continuación, se reflejan las pruebas llevadas a cabo para esta iteración del proyecto.

Prueba de Validación PV-1: Login		
Opciones	**Descripción**	**Salida**
Correcto	Accede con el usuario y contraseña provisional para desarrollo	El usuario accede correctamente a la página principal de la aplicación.
Incorrecto	Accede con otras credenciales diferentes a las definidas para el desarrollo	Se muestra mensaje de error de credenciales incorrectas.

Esta prueba se ha realizado sobre el módulo provisional de autenticación, ya que en esta iteración no se ha desarrollado la conexión con el servidor y la base de datos del centro médico.

Prueba de Validación PV-2: Crear Escala		
Opciones	**Descripción**	**Salida**
Número Positivo	Genera una escala de evaluación con 4 tareas.	Se generan las tablas de recogida de datos correctamente
Número Negativo	Genera una escala de evaluación con -2 tareas.	No se genera ninguna tabla

El funcionamiento es correcto.

Prueba de Validación PV-3: Evalúa		
Opciones	**Descripción**	**Salida**
Seleccionar campos de la escala de evaluación correctamente	Usuario selecciona los campos deseados de la escala. Uno por tarea.	Se muestra el diagnóstico según la escala de evaluación cumplimentada.
Selecciona varias opciones para una misma tarea	Usuario selecciona varias opciones de una misma tarea.	Se muestra el diagnóstico sin tener en cuenta que no se puede seleccionar varias opciones para una misma tarea.

La prueba PV-3 refleja que el proceso de evaluación debe contemplar y no permitir que el usuario pulse varias opciones para una misma tarea. Esto no debería permitirse. Se debe subsanar en siguientes iteraciones.

Prueba de Validación PV-4: Lista de Escalas de Evaluación		
Opciones	**Descripción**	**Salida**
Acceder a la lista de escalas de evaluación	Usuario accede a la pantalla de lista de escalas de evaluación.	Se muestra correctamente una lista con las escalas de evaluación disponibles.

Funcionamiento es correcto.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Este trabajo me ha permitido poner a prueba muchos de mis conocimientos pero el aspecto más importante ha sido el de trabajar sobre un guión elaborado por mi mismo. Esto ha sido difícil en muchas ocasiones.

El comienzo fue complicado debido a que este trabajo de fin de grado fue propuesto por mi. Tenía que definir bien el alcance del proyecto y cómo implementarlo. Gracias a mi tutor, pude definir bien hasta dónde quería llegar pero no sentía la seguridad que se siente en las prácticas de las asignaturas cursadas en el grado.

A la hora de decidir cómo implementarlo, elegí un framework que no me facilitó las cosas debido al carácter tan dinámico que demandaba esta aplicación. Esto puso a prueba todos mis conocimientos y me retrasó notablemente.

Ha sido una experiencia nueva de la que he aprendido mucho a nivel técnico. He trabajado con herramientas muy novedosas y en continua evolución que me ha llevado a buscar información en foros de todo el mundo. El proceso de búsqueda de información ha sido lo más complicado del proyecto sin duda. No hay manuales, se aprende a base de aportaciones de desarrolladores que, en muchas ocasiones, se quedan obsoletas con meses de antigüedad debido al vertiginoso ritmo con que evolucionan estas herramientas de desarrollo web.

Este proyecto ha sido un gran reto y me ha puesto a prueba tanto en conocimientos técnicos como en capacidad de organización.

En lo referente al proyecto, creo que he conseguido una aplicación sencilla de utilizar y con una estética muy agradable. En mi opinión, tiene un largo recorrido aún y podría ser un buen proyecto de negocio si se decidiera llevarlo a cabo. La mayor complejidad considero que se encuentra en el tratamiento de datos médicos, un proceso muy delicado.

6.2 Trabajo futuro

Para el trabajo futuro, a parte de los requisitos que no se han implementado en esta primera iteración de desarrollo del proyecto, se proponen varias mejoras.

- Guardado del formulario de cada escala de evaluación con la cumplimetación del profesional. No solo el diagnóstico como se hace en el proyecto actual.
- Establecimiento de conexiones con el servidor del centro médico para guardar los resultados en su base de datos.
- Asegurar que los datos médicos cumplen la normativa de manipulación de datos médicos.
- Elaboración una red de comunicación entre centros que utilicen la aplicación para compartir escalas de evaluación.

- Migración de la aplicación a servicios cloud como los de Amazon, Google o Microsoft para aumentar la eficiencia y rendimiento.

Referencias

[1] Escala Barthel
https://es.wikipedia.org/wiki/%C3%8Dndice_de_Barthel
[2] Escala de Lawton y Brody
https://www.hipocampo.org/lawton-brody.asp
[3] Escala Tinetti
http://www.infodoctor.org/www/escalatinetti.htm
[4] Escalas de Valoración
http://www.grupoeetph.com/content/uploads/2016/11/Escalas-Valoracio%CC%81n-Carol-Bonilla-HCB-1-1.pdf
[5] Los 5 mejores Frameworks javascript de 2017
https://www.campusmvp.es/recursos/post/los-5-mejores-frameworks-de-javascript-en-2017.aspx
[6] Frameworks Front-End de 2017 para grandes empresas
http://www.fecron.com/frameworks-front-end-para-grandes-empresas/
[7] Angular
https://es.wikipedia.org/wiki/Angular_(framework)
[8] React
https://es.wikipedia.org/wiki/React
[9] ¿Por qué tiene tanto éxito React?
https://www.arsys.es/blog/programacion/react-javascript/
[10] ¿Por qué tiene tanto éxito React?
https://www.arsys.es/blog/programacion/react-javascript/
[11] Vue
https://en.wikipedia.org/wiki/Vue.js
[12] Introducción
https://es-vuejs.github.io/vuejs.org/v2/guide/
[13] Comparision with other frameworks
https://es-vuejs.github.io/vuejs.org/v2/guide/comparison.html
[14] Adrián Alonso Vega, ¿Por qué Vue.js está ganando tanta popularidad?,1, 7 Enero 2018
https://es-vuejs.github.io/vuejs.org/v2/guide/
[15] Vue vs React
https://es-vuejs.github.io/vuejs.org/v2/guide/comparison.html
[16] Bootstrap
https://es.wikipedia.org/wiki/Bootstrap_(framework)
[17] Amazon Web Services
https://aws.amazon.com/es/
[18] S3
https://aws.amazon.com/es/s3/
[19] EBS
https://aws.amazon.com/es/ebs/

[20] Google Cloud Services
https://cloud.google.com/
[21] Juan María Fiz, Google Cloud,1, 16 Noviembre 2016
https://www.paradigmadigital.com/techbiz/aws-vs-google-cloud-12-procesamiento-almacenamiento/
[22] Html
https://es.wikipedia.org/wiki/HTML
[23] Typescript
https://es.wikipedia.org/wiki/TypeScript
[24] Ng2-smart-table
https://akveo.github.io/ng2-smart-table/#/examples/using-filters
[25] Ng-editable-table
https://www.npmjs.com/package/ng-editable-table
[26] Bootswatch
https://bootswatch.com/materia/
[27] Figura 1: Frontend y Backend
https://www.techsuitenyc.com/front-end-vs-back-end-explained-non-technical-entrepreneurs/
[28] Figura 5: MVC Vue
https://adrianalonso.es/desarrollo-web/framework-js/por-que-vue-js-esta-ganando-tanta-popularidad/
[29] Figura 7: Amazon vs Google Fuente:
https://cloudacademy.com/blog/google-cloud-vs-aws-a-comparison/
[30] Figura 8: Escala Barthel
https://ximeromeroguiamp3.wordpress.com/2017/03/12/indice-de-barthel/
[31] Figura 9: Escala Barthel Resultados
https://ximeromeroguiamp3.wordpress.com/2017/03/12/indice-de-barthel/

Glosario

API	Application Programming Interface
Framework	Entorno o marco de trabajo
Base de datos	Estructura de almacenamiento de datos
Escala de Evaluación	Conjunto de cuestionarios que el observador cumplimenta anotando las conductas que observa.
Login	Iniciar sesión en una aplicación mediante credenciales
Logout	Salir de la sesión de un aplicación
RSA	Algoritmo de cifrado
Responsive	Diseño que adapta la vista al dispositivo que reproduce una aplicación
Javascript	Lenguaje de programación
Json	Formato de texto ligero para el intercambio de datos
Terapeuta Ocupacional	Profesionales sociosanitarios que dirigen su actuación a personas que presentan una discapacidad.
Servicios Cloud	Paradigma que permite ofrecer servicios de computación a través de una red
Servicios en la nube	Paradigma que permite ofrecer servicios de computación a través de una red
Migración	Proceso que necesitamos hacer para transferir los datos de un sistema a otro
Digital	Implica que los contenidos sólo pueden leerse con algún dispositivo digital o electrónico en lugar de estar impresos sobre el papel
Salud Mental	Estado de equilibrio entre una persona y su entorno socio-cultural
Geriatría	Parte de la medicina que se ocupa de las enfermedades propias de la vejez.
Apliación web	Herramientas que los usuarios pueden utilizar accediendo a un servidor web a través de internet o de una intranet mediante un navegador

Dispositivo móvil	Computadora de bolsillo o computadora de mano
SPA	Single Page Application. Aplicación de una sola página.
Logotipo	Signo fráfico que identifica una entidad.
NodeJS-Node	Servidor Web.
Máquinas virtuales	Software que simula un sistema de computación y puede ejecutar programas como si fuese una computadora real.
Módulos	Parte de código de un programa que ejecuta una funcionalidad concreta.
Componente	Cada entidad en que se divide una aplicación.
IDE	Entorno de desarrollo.
Extensiones	Complemento que añade una funcionalidad a un entorno de programación.
Testing	Pruebas realizadas a un programa.
Lenguaje declarativo	Lenguaje basado en el desarrollo de programas especificando un conjunto de condiciones, proposiciones, afirmaciones, restricciones, ecuaciones o transformaciones que describen el problema y detallan su solución.
Posicionamiento web	Visibilidad de una página web en la red.
Isomorfismo	Construir modelos similares al modelo original, esto con el fin de aumentar o mejorar el desempeño de un sistema.
Servidor web	Software que se ejecuta en un PC y se mantiene a la espera de recibir peticiones por parte de los usuarios de Internet.
MVC	Modelo Vista Controlador
UI	Interfaz de Usuario
Hardware	Conjunto de elementos físicos o materiales que constituyen una computadora o un sistema informático
Paquete npm	Encapsula un módulo Javascript
Librería	Módulo javascript que implementa una función concreta

Anexos